Amélioration de performances du solveur SMT Alt-Ergo grâce à l'intégration d'un solveur SAT efficace

Soutenance de thèse de doctorat

Albin Coquereau

École Nationale Supérieure des Techniques Avancées — OCamlPro

Sous la supervision académique de Michel Mauny et Sylvain Conchon

Sous la supervision industrielle de Mohamed Iguernlala et Fabrice Le Fessant

16 décembre 2019

Passage à niveau

- Entièrement automatisé
- Peut être bogué

Introduction

Prouver que les programmes respectent les spécifications

if *Train détecté* **then**
  activer les signaux lumineux
  activer les signaux sonores
  baisser les barrières après 10 secondes

Introduction

Prouver que les programmes respectent les spécifications

if *Train détecté* **then**
  activer les signaux lumineux
  activer les signaux sonores
  baisser les barrières après 10 secondes

Code + Spécification → Formules mathématiques → Atelier de preuve

- Quantité importante de formules produites
- La preuve manuelle est complexe, longue, fastidieuse et coûteuse

Introduction

Prouver que les programmes respectent les spécifications

if *Train détecté* **then**
    activer les signaux lumineux
    activer les signaux sonores
    baisser les barrières après 10 secondes

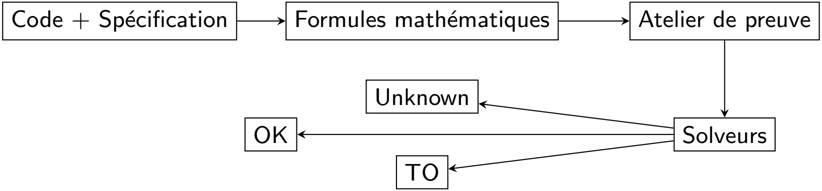
$$\boxed{\text{Code + Spécification}} \longrightarrow \boxed{\text{Formules mathématiques}} \longrightarrow \boxed{\text{Atelier de preuve}}$$

- Quantité importante de formules produites
- La preuve manuelle est complexe, longue, fastidieuse et coûteuse
- Besoin d'automatiser

Introduction

Prouver que les programmes respectent les spécifications

if *Train détecté* **then**
- activer les signaux lumineux
- activer les signaux sonores
- baisser les barrières après 10 secondes



- Quantité importante de formules produites
- La preuve manuelle est complexe, longue, fastidieuse et coûteuse
- Besoin d'automatiser

Le solveur Alt-Ergo

Le solveur Alt-Ergo

Alt-Ergo est un solveur de Satisfiabilité Modulo Théories, développé en OCaml

- Développé au LRI à partir de 2006
- Maintenu en partenariat avec OCamlPro depuis 2013

Le solveur Alt-Ergo

Alt-Ergo est un solveur de Satisfiabilité Modulo Théories, développé en OCaml

- Développé au LRI à partir de 2006
- Maintenu en partenariat avec OCamlPro depuis 2013

- Spécialisé dans la logique du premier ordre avec théories prédéfinies
- Unique solveur à supporter nativement du polymorphisme
  - Quantifier sur des variables types en plus de quantifier sur des variables de termes

```
type 'a t

logic P: int -> prop
logic Q: 'a t -> prop

axiom a1: forall x:int. P(x) -> (forall y:'a t. Q(y))

goal g1:
    forall z:'a t. P(1) -> Q(z)
```

Les utilisations d'Alt-Ergo

Utilisé par des plates-formes de preuve déductive telles que :

- Atelier-B (méthode B)
- Frama-C (C)
- Spark (Ada)
- Why3 (WhyML)

Il est aussi utilisé au sein du vérificateur de modèle Cubicle

Alt-Ergo et la preuve de programme

Résultats sur des bancs de test issus de la preuve de programme

- Limite de temps de 60 secondes
- Limite de mémoire de 2 Go

Alt-Ergo et la preuve de programme

Résultats sur des bancs de test issus de la preuve de programme

- Limite de temps de 60 secondes
- Limite de mémoire de 2 Go

	#buts	`Alt-Ergo`	`CVC4`	`Vampire`	`Z3`
BWARE–DAB	860	**100% (417s)**	94.5%(1295s)	46.6%(3040s)	81.9%(128s)
BWARE–RCS3	2256	**98.9% (685s)**	90.4% (2845s)	47.6%(455s)	97.1%(229s)
BWARE–p4	9351	**99.3% (2279s)**	97.4%(9754s)	32.5%(24878s)	85.1%(1170s)
BWARE–p9	371	**67.9% (342s)**	50.4%(1018s)	14.8%(1426s)	64.2%(464s)
EACSL	959	**93.3% 238s**	93.3% (388s)	45.3%(4999s)	77.6% (619s)
SPARK	16773	83.6% (2757s)	85.1%(558s)	72.9%(16725s)	**90.3%(1545s)**
WHY3	2003	**72.0% (1876s)**	65.9%(632s)	31.1%(6301s)	59.7%(1429s)
Total	32563	**89.0%(6119s)**	87.9%(16492s)	54.8%(57827s)	86.6%(5586s)

Alt-Ergo et la preuve de programme

Résultats sur des bancs de test issus de la preuve de programme

- Limite de temps de 60 secondes
- Limite de mémoire de 2 Go

	#buts	Alt-Ergo	CVC4	Vampire	Z3
BWARE–DAB	860	100% (417s)	94.5%(1295s)	46.6%(3040s)	81.9%(128s)
BWARE–RCS3	2256	98.9% (685s)	90.4% (2845s)	47.6%(455s)	97.1%(229s)
BWARE–p4	9351	99.3% (2279s)	97.4%(9754s)	32.5%(24878s)	85.1%(1170s)
BWARE–p9	371	67.9% (342s)	50.4%(1018s)	14.8%(1426s)	64.2%(464s)
EACSL	959	93.3% 238s	93.3% (388s)	45.3%(4999s)	77.6% (619s)
SPARK	16773	83.6% (2757s)	85.1%(558s)	72.9%(16725s)	90.3%(1545s)
WHY3	2003	72.0% (1876s)	65.9%(632s)	31.1%(6301s)	59.7%(1429s)
Total	32563	89.0%(6119s)	87.9%(16492s)	54.8%(57827s)	86.6%(5586s)

- Peu de raisonnement booléen (sauf SPARK)

Problèmatique

Améliorer les performances du solveur SAT d'Alt-Ergo pour

- Améliorer les performances globales
- Pouvoir supporter de nouveaux problèmes nécessitant un traitement SAT efficace
  - Vecteurs de bits
  - Entiers machine (32bits/64bits)

Problèmatique

Améliorer les performances du solveur SAT d'Alt-Ergo pour

- Améliorer les performances globales
- Pouvoir supporter de nouveaux problèmes nécessitant un traitement SAT efficace
  - Vecteurs de bits
  - Entiers machine (32bits/64bits)

	# buts	`Alt-Ergo`	
SATLIB	1595	1.8 %(535s)	

Problèmatique

Améliorer les performances du solveur SAT d'Alt-Ergo pour

- Améliorer les performances globales
- Pouvoir supporter de nouveaux problèmes nécessitant un traitement SAT efficace
  - Vecteurs de bits
  - Entiers machine (32bits/64bits)

	# buts	`Alt-Ergo`	`satml`
SATLIB	1595	1.8 %(535s)	**99.3% (1962s)**

- `satml` ré-implémentation de `minisat` en OCaml

Travaux préliminaires

Intégration de façon naïve d'un solveur SAT performant au sein d'Alt-Ergo :

Travaux préliminaires

Intégration de façon naïve d'un solveur SAT performant au sein d'Alt-Ergo :

- `tableaux` : solveur SAT historique d'Alt-Ergo

	# buts	`Alt-Ergo (satml)`	`Alt-Ergo (tableaux)`
BWARE–DAB	860	98.7% (258s)	**100% (417s)**
BWARE–RCS3	2256	98.7% (742s)	**98.9% (685s)**
BWARE–p4	9341	98.4% (2097s)	**99.3% (2279s)**
BWARE–p9	371	64.7% (1104s)	**67.9% (342s)**
EACSL	959	75.6% (64s)	**93.3% (258s)**
SPARK	16773	80.5% (1769s)	**83.6% (2757s)**
WHY3	2003	38.9% (616s)	**72.0% (1876s)**
Total	32563	84.5% (6652s)	**89.0% (8617s)**

Travaux préliminaires

Intégration de façon naïve d'un solveur SAT performant au sein d'Alt-Ergo :

- `tableaux` : solveur SAT historique d'Alt-Ergo

	# buts	`Alt-Ergo (satml)`	`Alt-Ergo (tableaux)`
BWARE–DAB	860	98.7% (258s)	**100% (417s)**
BWARE–RCS3	2256	98.7% (742s)	**98.9% (685s)**
BWARE–p4	9341	98.4% (2097s)	**99.3% (2279s)**
BWARE–p9	371	64.7% (1104s)	**67.9% (342s)**
EACSL	959	75.6% (64s)	**93.3% (258s)**
SPARK	16773	80.5% (1769s)	**83.6% (2757s)**
WHY3	2003	38.9% (616s)	**72.0% (1876s)**
Total	32563	84.5% (6652s)	**89.0% (8617s)**

Utiliser un CDCL efficace n'est pas suffisant pour améliorer les performances de notre solveur SMT

Contributions de cette thèse

- Intégration efficace d'un solveur SAT performant au sein du solveur SMT Alt-Ergo

Contributions de cette thèse

- Intégration efficace d'un solveur SAT performant au sein du solveur SMT Alt-Ergo
- Extension de PSMT2 et son intégration dans Alt-Ergo

Contributions de cette thèse

- Intégration efficace d'un solveur SAT performant au sein du solveur SMT Alt-Ergo
- Extension de PSMT2 et son intégration dans Alt-Ergo
- Participation à la compétition SMT-COMP

Contributions de cette thèse

- Intégration efficace d'un solveur SAT performant au sein du solveur SMT Alt-Ergo
- Extension de PSMT2 et son intégration dans Alt-Ergo
- Participation à la compétition SMT-COMP
- Étude comparative de performance entre deux implémentations en C++ et en OCaml du solveur SAT de référence, MiniSat

Intégration efficace d'un solveur SAT performant au sein du solveur SMT Alt-Ergo

- Problème SAT
- Satisfiabilité modulo théories
- Le solveur SAT d'Alt-Ergo
- Solution d'intégration efficace

Problème SAT

$$(A \Rightarrow B) \wedge (C \Rightarrow D) \wedge (\neg E \vee (\neg F \wedge (F \vee \neg B))) \wedge (E \vee (G \wedge (\neg G \vee \neg B)))$$

Problème SAT

$$(A \Rightarrow B) \wedge (C \Rightarrow D) \wedge (\neg E \vee (\neg F \wedge (F \vee \neg B))) \wedge (E \vee (G \wedge (\neg G \vee \neg B)))$$

Modèle booléen renvoyé par le solveur SAT :
$\{\neg A;\ \neg B;\ \neg C;\ D;\ \neg E;\ F;\ G\}$

Problème SAT

$$(A \Rightarrow B) \wedge (C \Rightarrow D) \wedge (\neg E \vee (\neg F \wedge (F \vee \neg B))) \wedge (E \vee (G \wedge (\neg G \vee \neg B)))$$

Modèle booléen renvoyé par le solveur SAT :
$\{\neg A;\ \neg B;\ \neg C;\ D;\ \neg E;\ F;\ G\}$

Technique de recherche de modèle

- Algorithme de backtracking
- Arbre de décision
- Propagation de Contraintes Booléennes
- Formule en Forme Normale Conjonctive

Algorithme DPLL (1962)

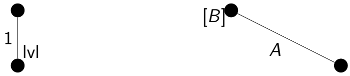
$$\begin{aligned} & (\neg A \vee B) \\ \wedge\ & (\neg C \vee D) \\ \wedge\ & (\neg E \vee \neg F) \\ \wedge\ & (F \vee \neg E \vee \neg B) \\ \wedge\ & (E \vee G) \\ \wedge\ & (E \vee \neg G \vee \neg B) \end{aligned}$$

Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$

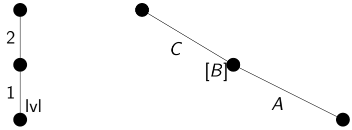
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
& \wedge (\neg C \vee D) \\
& \wedge (\neg E \vee \neg F) \\
& \wedge (F \vee \neg E \vee \neg B) \\
& \wedge (E \vee G) \\
& \wedge (E \vee \neg G \vee \neg B)
\end{aligned}
$$



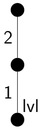
Algorithme DPLL (1962)

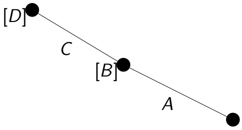
$$\begin{aligned} & (\neg A \vee B) \\ \wedge\ & (\neg C \vee D) \\ \wedge\ & (\neg E \vee \neg F) \\ \wedge\ & (F \vee \neg E \vee \neg B) \\ \wedge\ & (E \vee G) \\ \wedge\ & (E \vee \neg G \vee \neg B) \end{aligned}$$



Algorithme DPLL (1962)

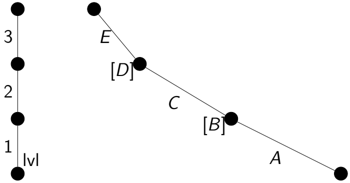
$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$





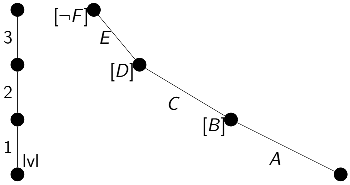
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$



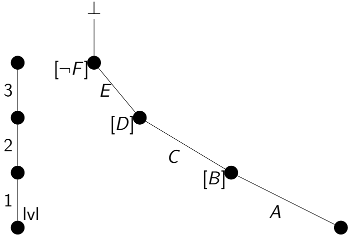
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$



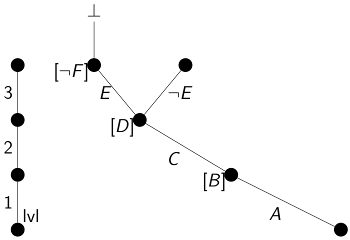
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$



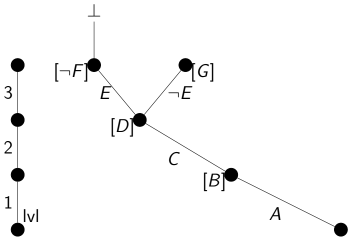
Algorithme DPLL (1962)

$$\begin{aligned} & (\neg A \vee B) \\ \wedge\ & (\neg C \vee D) \\ \wedge\ & (\neg E \vee \neg F) \\ \wedge\ & (F \vee \neg E \vee \neg B) \\ \wedge\ & (E \vee G) \\ \wedge\ & (E \vee \neg G \vee \neg B) \end{aligned}$$



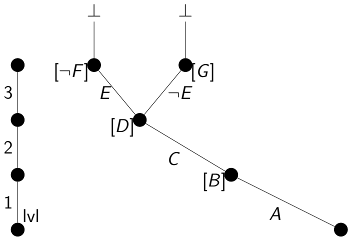
Algorithme DPLL (1962)

$$
\begin{aligned}
&(\neg A \lor B)\\
\land\ &(\neg C \lor D)\\
\land\ &(\neg E \lor \neg F)\\
\land\ &(F \lor \neg E \lor \neg B)\\
\land\ &(E \lor G)\\
\land\ &(E \lor \neg G \lor \neg B)
\end{aligned}
$$



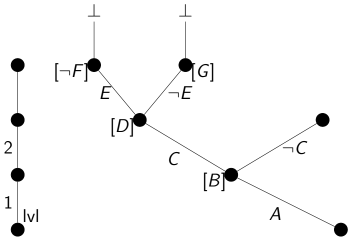
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$



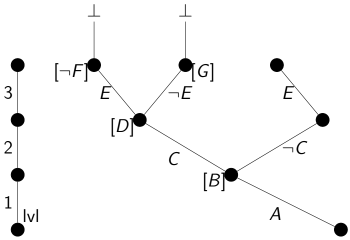
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$



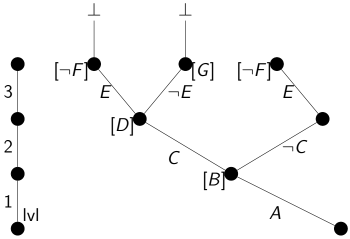
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \lor B) \\
\land\ & (\neg C \lor D) \\
\land\ & (\neg E \lor \neg F) \\
\land\ & (F \lor \neg E \lor \neg B) \\
\land\ & (E \lor G) \\
\land\ & (E \lor \neg G \lor \neg B)
\end{aligned}
$$



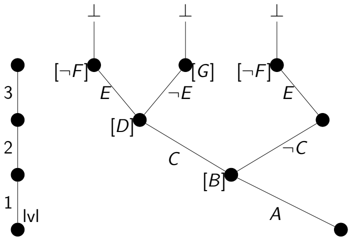
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$



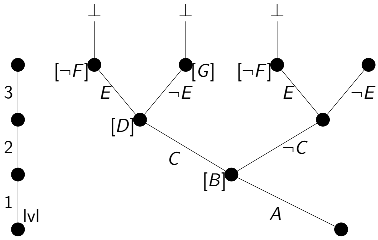
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$



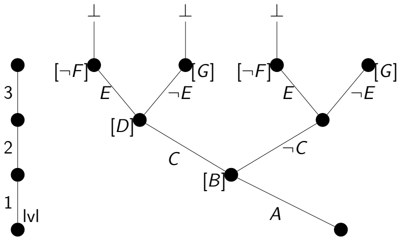
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$



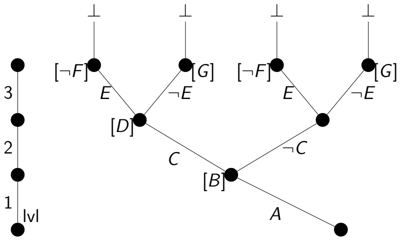
Algorithme DPLL (1962)

$$
\begin{aligned}
&(\neg A \vee B) \\
\wedge\ &(\neg C \vee D) \\
\wedge\ &(\neg E \vee \neg F) \\
\wedge\ &(F \vee \neg E \vee \neg B) \\
\wedge\ &(E \vee G) \\
\wedge\ &(E \vee \neg G \vee \neg B)
\end{aligned}
$$



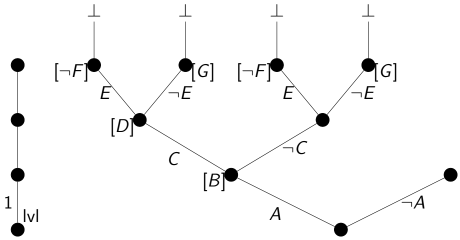
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & \color{red}{(E \vee \neg G \vee \neg B)}
\end{aligned}
$$



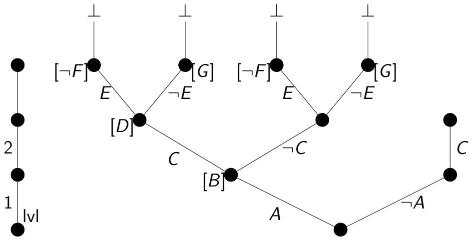
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$



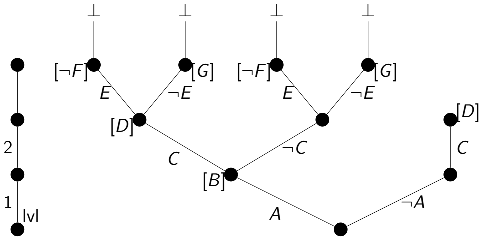
Algorithme DPLL (1962)

$$
\begin{aligned}
&(\neg A \vee B)\\
\wedge\ &(\neg C \vee D)\\
\wedge\ &(\neg E \vee \neg F)\\
\wedge\ &(F \vee \neg E \vee \neg B)\\
\wedge\ &(E \vee G)\\
\wedge\ &(E \vee \neg G \vee \neg B)
\end{aligned}
$$



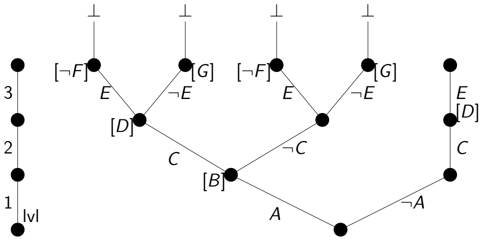
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$



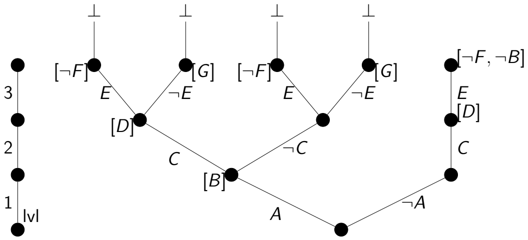
Algorithme DPLL (1962)

$$
\begin{aligned}
&(\neg A \vee B) \\
\wedge\ &(\neg C \vee D) \\
\wedge\ &(\neg E \vee \neg F) \\
\wedge\ &(F \vee \neg E \vee \neg B) \\
\wedge\ &(E \vee G) \\
\wedge\ &(E \vee \neg G \vee \neg B)
\end{aligned}
$$



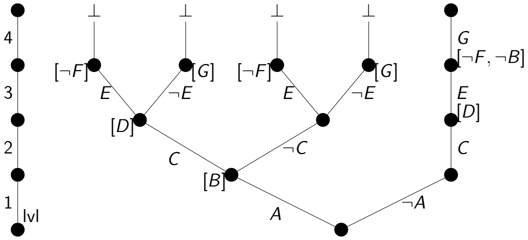
Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\ & (\neg C \vee D) \\
\wedge\ & (\neg E \vee \neg F) \\
\wedge\ & (F \vee \neg E \vee \neg B) \\
\wedge\ & (E \vee G) \\
\wedge\ & (E \vee \neg G \vee \neg B)
\end{aligned}
$$



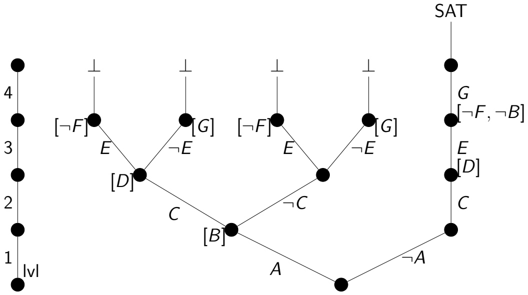
Algorithme DPLL (1962)

$$
\begin{aligned}
&(\neg A \vee B) \\
\wedge\ &(\neg C \vee D) \\
\wedge\ &(\neg E \vee \neg F) \\
\wedge\ &(F \vee \neg E \vee \neg B) \\
\wedge\ &(E \vee G) \\
\wedge\ &(E \vee \neg G \vee \neg B)
\end{aligned}
$$



Algorithme DPLL (1962)

$$
\begin{aligned}
& (\neg A \lor B) \\
\land\ & (\neg C \lor D) \\
\land\ & (\neg E \lor \neg F) \\
\land\ & (F \lor \neg E \lor \neg B) \\
\land\ & (E \lor G) \\
\land\ & (E \lor \neg G \lor \neg B)
\end{aligned}
$$



Algorithme DPLL (1962)

Input: Γ : CNF, Δ: Boolean Model
Output: Satisfiability status

Algorithme DPLL (1962)

Input: Γ : CNF, Δ: Boolean Model
Output: Satisfiability status

```
lvl ← 0
while true do
```

Algorithme DPLL (1962)

Input: Γ : CNF, Δ: Boolean Model
Output: Satisfiability status

```
lvl ← 0
while true do
    (Δ, Conflict) ← BCP(Γ, Δ)
```

Algorithme DPLL (1962)

Input: Γ : CNF, Δ: Boolean Model
Output: Satisfiability status

```
lvl ← 0
while true do
    (Δ, Conflict) ← BCP(Γ, Δ)
    if Conflict ≠ ∅ then
```

Algorithme DPLL (1962)

Input: Γ : CNF, Δ: Boolean Model
Output: Satisfiability status

```
lvl ← 0
while true do
    (Δ, Conflict) ← BCP(Γ, Δ)
    if Conflict ≠ ∅ then
        if lvl = 0 then
            return UNSAT
```

Algorithme DPLL (1962)

```
Input: Γ : CNF, Δ: Boolean Model
Output: Satisfiability status
lvl ← 0
while true do
    (Δ, Conflict) ← BCP(Γ, Δ)
    if Conflict ≠ ∅ then
        if lvl = 0 then
            return UNSAT
        else
            lvl ← lvl − 1
            Δ ← backtrack(Γ, Δ)
```

Algorithme DPLL (1962)

```
Input: Γ : CNF, Δ: Boolean Model
Output: Satisfiability status
lvl ← 0
while true do
    (Δ, Conflict) ← BCP(Γ, Δ)
    if Conflict ≠ ∅ then
        if lvl = 0 then
            return UNSAT
        else
            lvl ← lvl − 1
            Δ ← backtrack(Γ, Δ)
    else if all variables are assigned in Δ then
        return SAT
```

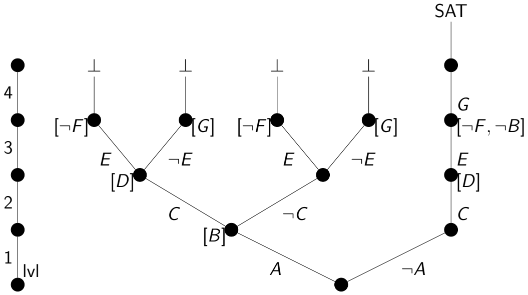
Algorithme DPLL (1962)

Input: Γ : CNF, Δ: Boolean Model
Output: Satisfiability status

$lvl \leftarrow 0$
while *true* **do**
    $(\Delta, Conflict) \leftarrow BCP(\Gamma, \Delta)$
    if $Conflict \neq \emptyset$ **then**
        if $lvl = 0$ **then**
            return UNSAT
        else
            $lvl \leftarrow lvl - 1$
            $\Delta \leftarrow backtrack(\Gamma, \Delta)$
    else if *all variables are assigned in* Δ **then**
        return SAT
    else
        $L \leftarrow choose(\Gamma, \Delta)$
        $lvl \leftarrow lvl + 1$
        $\Delta \leftarrow L :: \Delta$

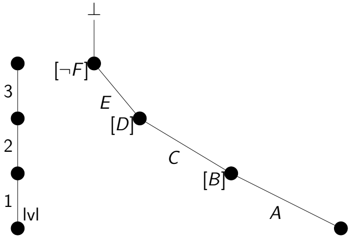
Algorithme DPLL (1962)

$$
\begin{aligned}
&(\neg A \lor B) \\
\land\, &(\neg C \lor D) \\
\land\, &(\neg E \lor \neg F) \\
\land\, &(F \lor \neg E \lor \neg B) \\
\land\, &(E \lor G) \\
\land\, &(E \lor \neg G \lor \neg B)
\end{aligned}
$$



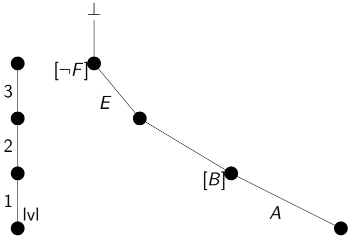
Algorithme CDCL (1996)

$$
\begin{aligned}
&(\neg A \vee B) \\
\wedge\, &(\neg C \vee D) \\
\wedge\, &(\neg E \vee \neg F) \\
\wedge\, &(F \vee \neg E \vee \neg B) \\
\wedge\, &(E \vee G) \\
\wedge\, &(E \vee \neg G \vee \neg B)
\end{aligned}
$$



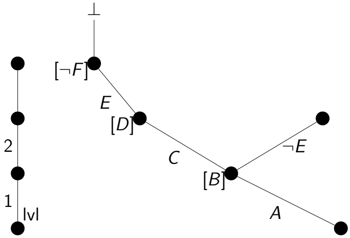
Algorithme CDCL (1996)

$$
\begin{aligned}
&(\neg A \vee B) \\
\wedge\, &(\neg C \vee D) \\
\wedge\, &(\neg E \vee \neg F) \\
\wedge\, &(F \vee \neg E \vee \neg B) \\
\wedge\, &(E \vee G) \\
\wedge\, &(E \vee \neg G \vee \neg B)
\end{aligned}
$$



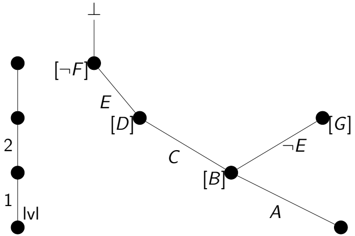
Algorithme CDCL (1996)

$$
\begin{aligned}
&(\neg A \lor B) \\
\land\, &(\neg C \lor D) \\
\land\, &(\neg E \lor \neg F) \\
\land &(F \lor \neg E \lor \neg B) \\
\land &(E \lor G) \\
\land &(E \lor \neg G \lor \neg B) \\
\land &(\neg A \lor \neg E)
\end{aligned}
$$



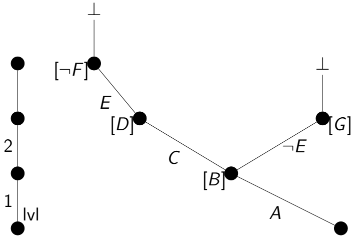
Algorithme CDCL (1996)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\, & (\neg C \vee D) \\
\wedge\, & (\neg E \vee \neg F) \\
\wedge & (F \vee \neg E \vee \neg B) \\
\wedge & (E \vee G) \\
\wedge & (E \vee \neg G \vee \neg B) \\
\wedge & (\neg A \vee \neg E)
\end{aligned}
$$



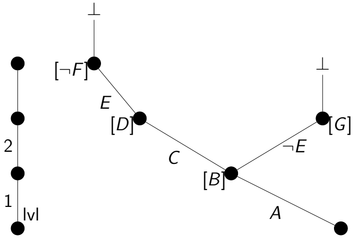
Algorithme CDCL (1996)

$$
\begin{aligned}
& (\neg A \vee B) \\
\wedge\, & (\neg C \vee D) \\
\wedge\, & (\neg E \vee \neg F) \\
\wedge\, & (F \vee \neg E \vee \neg B) \\
\wedge\, & (E \vee G) \\
\wedge\, & (E \vee \neg G \vee \neg B) \\
\wedge\, & (\neg A \vee \neg E)
\end{aligned}
$$



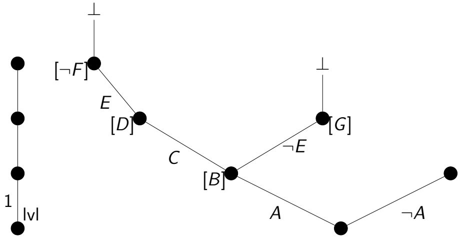
Algorithme CDCL (1996)

$$
\begin{aligned}
&(\neg A \vee B) \\
\wedge\, &(\neg C \vee D) \\
\wedge\, &(\neg E \vee \neg F) \\
\wedge\, &(F \vee \neg E \vee \neg B) \\
\wedge\, &(E \vee G) \\
\wedge\, &(E \vee \neg G \vee \neg B) \\
\wedge\, &(\neg A \vee \neg E) \\
\wedge\, &(\neg A)
\end{aligned}
$$



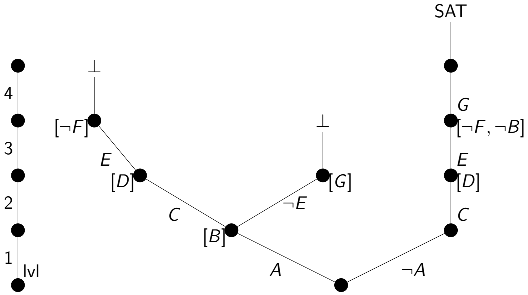
Algorithme CDCL (1996)

$$
\begin{aligned}
&(\neg A \vee B)\\
\wedge\, &(\neg C \vee D)\\
\wedge\, &(\neg E \vee \neg F)\\
\wedge &(F \vee \neg E \vee \neg B)\\
\wedge &(E \vee G)\\
\wedge &(E \vee \neg G \vee \neg B)\\
\wedge &(\neg A \vee \neg E)\\
\wedge &(\neg A)
\end{aligned}
$$



Algorithme CDCL (1996)

$$
\begin{aligned}
&(\neg A \vee B) \\
\wedge\,&(\neg C \vee D) \\
\wedge\,&(\neg E \vee \neg F) \\
\wedge\,&(F \vee \neg E \vee \neg B) \\
\wedge\,&(E \vee G) \\
\wedge\,&(E \vee \neg G \vee \neg B) \\
\wedge\,&(\neg A \vee \neg E) \\
\wedge\,&(\neg A)
\end{aligned}
$$



Algorithme CDCL (1996)

```
while true do
    (Δ, Conflict) ← BCP(Γ, Δ)
    if Conflict ≠ ∅ then
        if lvl = 0 then
            return UNSAT
        else
            (L ∨ C, bj_lvl) ← resolve(Γ, Δ, Conflict)
    else if all variables are assigned in Δ then
        return SAT
    else
        L ← choose(Γ, Δ)
        lvl ← lvl + 1
        Δ ← L :: Δ
```

Algorithme CDCL (1996)

while *true* **do**
- $(\Delta, Conflict) \leftarrow BCP(\Gamma, \Delta)$
- **if** $Conflict \neq \emptyset$ **then**
  - **if** $lvl = 0$ **then**
    - **return** UNSAT
  - **else**
    - $(L \vee C, bj\_lvl) \leftarrow resolve(\Gamma, \Delta, Conflict)$
    - $\Gamma \leftarrow \Gamma \cup \{L \vee C\}$
- **else if** *all variables are assigned in* Δ **then**
  - **return** SAT
- **else**
  - $L \leftarrow choose(\Gamma, \Delta)$
  - $lvl \leftarrow lvl + 1$
  - $\Delta \leftarrow L :: \Delta$

Algorithme CDCL (1996)

while *true* **do**
    $(\Delta, \mathit{Conflict}) \leftarrow \mathit{BCP}(\Gamma, \Delta)$
    if $\mathit{Conflict} \neq \emptyset$ **then**
        if $\mathit{lvl} = 0$ **then**
            return UNSAT
        else
            $(L \vee C, \mathit{bj\_lvl}) \leftarrow \mathit{resolve}(\Gamma, \Delta, \mathit{Conflict})$
            $\Gamma \leftarrow \Gamma \cup \{L \vee C\}$
            $(\Delta, \mathit{lvl}) \leftarrow \mathit{backjump}(\Delta, \mathit{bj\_lvl})$
    else if *all variables are assigned in* Δ **then**
        return SAT
    else
        $L \leftarrow \mathit{choose}(\Gamma, \Delta)$
        $\mathit{lvl} \leftarrow \mathit{lvl} + 1$
        $\Delta \leftarrow L :: \Delta$

Algorithme CDCL (1996)

```
while true do
    (Δ, Conflict) ← BCP(Γ, Δ)
    if Conflict ≠ ∅ then
        if lvl = 0 then
            return UNSAT
        else
            (L ∨ C, bj_lvl) ← resolve(Γ, Δ, Conflict)
            Γ ← Γ ∪ {L ∨ C}
            (Δ, lvl) ← backjump(Δ, bj_lvl)
            Δ ← L :: Δ
    else if all variables are assigned in Δ then
        return SAT
    else
        L ← choose(Γ, Δ)
        lvl ← lvl + 1
        Δ ← L :: Δ
```

Problème SAT

- Recherche d'un modèle
  - DPLL
  - CDCL

Problème SAT

- Recherche d'un modèle
  - DPLL
  - CDCL
- Satisfiabilité Modulo Théories

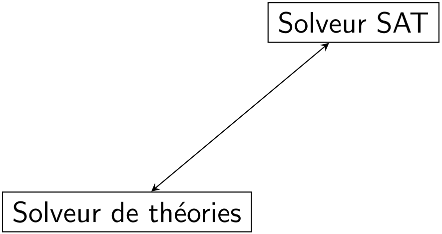
Satisfiabilité modulo théories

$$((x > 1) \vee ((\forall z.h(z) = 0) \wedge (h(2.0) \leq 0)))$$
$$\wedge ((x = 0) \vee (((\forall z.f(z) = -2z) \vee (x = y) \wedge (f(y) < 0)))$$

$$\boxed{\text{Solveur SAT}}$$

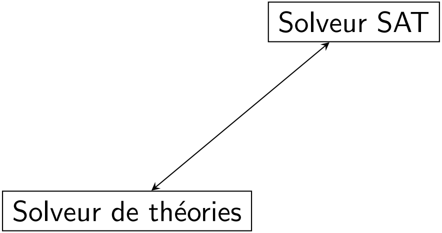
Satisfiabilité modulo théories

$$
\begin{gathered}
((x > 1) \vee ((\forall z. h(z) = 0) \wedge (h(2.0) \leq 0))) \\
\wedge ((x = 0) \vee (((\forall z. f(z) = -2z) \vee (x = y) \wedge (f(y) < 0)))
\end{gathered}
$$



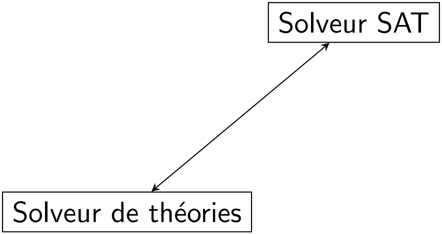
Satisfiabilité modulo théories

$$((x > 1) \vee ((\forall z. h(z) = 0) \wedge (h(2.0) \leq 0)))$$
$$\wedge\, ((x = 0) \vee (((\forall z. f(z) = -2z) \vee (x = y) \wedge (f(y) < 0)))$$



Satisfiabilité modulo théories

$$((x > 1) \lor ((\forall z. h(z) = 0) \land (h(2.0) \leq 0)))$$
$$\land ((x = 0) \lor (((\forall z. f(z) = -2z) \lor (x = y) \land (f(y) < 0)))$$



Satisfiabilité modulo théories

$$
\begin{aligned}
&((x > 1) \vee ((\forall z. h(z) = 0) \wedge (h(2.0) \leq 0))) \\
\wedge\, &((x = 0) \vee (((\forall z. f(z) = -2z) \vee {\color{blue}(x = y)} \wedge (f(y) < 0)))
\end{aligned}
$$

$$
\begin{tikzcd}
 & \boxed{\text{Solveur SAT}} \arrow[dl, leftrightarrow] \\
\boxed{\text{Solveur de théories}} &
\end{tikzcd}
$$

Interaction avec le solveur SAT

$$
\begin{gathered}
((x > 1) \vee (\forall z.h(z) = 0) \wedge (h(2.0) \leq 0))) \\
\wedge ((x = 0) \vee (((\forall z.f(z) = -2z) \vee (x = y) \wedge (f(y) < 0)))
\end{gathered}
$$

Introduction de littéraux :

$$
\begin{array}{llll}
A \iff x > 1 & B \iff \forall z.h(z) = 0 \\
C \iff h(2.0) \leq 0 & D \iff x = 0 \\
E \iff \forall z.f(z) = -2z & \\
F \iff x = y & G \iff f(y) < O
\end{array}
$$

Interaction avec le solveur SAT

$$((x > 1) \vee (\forall z.h(z) = 0) \wedge (h(2.0) \leq 0)))$$
$$\wedge ((x = 0) \vee (((\forall z.f(z) = -2z) \vee (x = y) \wedge (f(y) < 0)))$$

Introduction de littéraux :

$$\begin{array}{lll} A \iff x > 1 & & B \iff \forall z.h(z) = 0 \\ C \iff h(2.0) \leq 0 & & D \iff x = 0 \\ E \iff \forall z.f(z) = -2z & & \\ F \iff x = y & & G \iff f(y) < O \end{array}$$

Formule purement booléenne :

$$(A \vee (B \wedge C)) \wedge (D \vee ((E \vee F) \wedge G))$$

Interaction avec le solveur SAT

$$(A \vee (B \wedge C)) \wedge (D \vee ((E \vee F) \wedge G)$$

Modèle booléen renvoyé par le solveur SAT :
$\{A; \neg B; \neg C; D; E; F; G\}$

Interaction avec le solveur SAT

$$(A \vee (B \wedge C)) \wedge (D \vee ((E \vee F) \wedge G))$$

Modèle booléen renvoyé par le solveur SAT :
$\{A; \neg B; \neg C; D; E; F; G\}$

$$A \iff x > 1$$
$$D \iff x = 0$$

Interaction avec le solveur SAT

$$(A \vee (B \wedge C)) \wedge (D \vee ((E \vee F) \wedge G)$$

Modèle booléen renvoyé par le solveur SAT :
$\{A; \neg B; \neg C; D; E; F; G\}$

$$A \iff x > 1$$
$$D \iff x = 0$$

Ajout de la clause :

$$\neg A \vee \neg D$$

Interaction avec le solveur SAT

$$(A \vee (B \wedge C)) \wedge (D \vee ((E \vee F) \wedge G)$$

Modèle booléen renvoyé par le solveur SAT :
$\{A; \neg B; \neg C; D; E; F; G\}$

$$A \iff x > 1$$
$$D \iff x = 0$$

Ajout de la clause :

$$\neg A \vee \neg D$$

Modèle booléen renvoyé par le solveur SAT :
$\{A; \neg B; \neg C; \neg D; E; F; G\}$

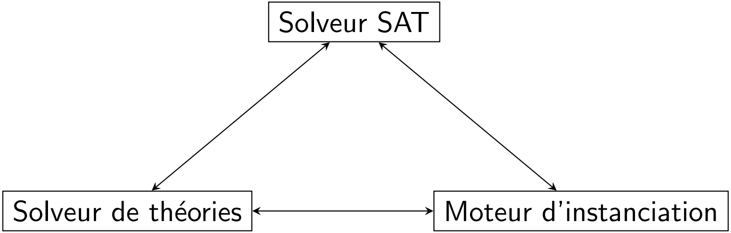
Modification du Solveur CDCL pour interagir avec le solveur de théories et le moteur d'instanciation, CDCL(T)

Algorithme CDCL(T)

```
while true do
    (Δ, Conflict) ← BCP(Γ, Δ)
    if Conflict ≠ ∅ then
        ...
    else
        (T, Conflict) ← theory_assume(Δ, T)
        if Conflict ≠ ∅ then
            if lvl = 0 then
                return UNSAT
            else
                (L∨C, bj_lvl) ← resolve(Γ, Δ, T, Conflict)
                Γ ← Γ ∪ {L ∨ C}
                (Δ, T, lvl) ← backjump(bj_lvl)
                Δ ← L :: Δ
        else if all variables are assigned in Δ then
            ...
```

Satisfiabilité modulo théories

$$
\begin{aligned}
&((x > 1) \lor ((\forall z. h(z) = 0) \land (h(2.0) \leq 0))) \\
\land\, &((x = 0) \lor (((\forall z. f(z) = -2z) \lor (x = y) \land (f(y) < 0)))
\end{aligned}
$$



Moteur d'instanciation

Générer de nouveaux faits à partir des formules quantifiées.

Moteur d'instanciation

Générer de nouveaux faits à partir des formules quantifiées.

$\forall x : int.P(f(g(x)))$

- $P(f(g(1)))$
- $P(f(g(2)))$
- $P(f(g(a)))$
- ...

Moteur d'instanciation

Générer de nouveaux faits à partir des formules quantifiées.

$\forall x : int.P(f(g(x)))$

- $P(f(g(1)))$
- $P(f(g(2)))$
- $P(f(g(a)))$
- ...

Principal défi :

- Ne pas générer trop d'instances pour ne pas saturer le solveur SMT

Moteur d'instanciation

Utilise un système de gardes/motifs pour limiter le nombre d'instances générées :

$\forall x : int.P(f(g(x)))$

- garde possible : $f(g(x))$

Moteur d'instanciation

Utilise un système de gardes/motifs pour limiter le nombre d'instances générées :

$\forall x : int.P(f(g(x)))$

- garde possible : $f(g(x))$

Contexte des théories :

- $f(1)$
- $g(2)$
- $\boldsymbol{f(g(a))}$

Moteur d'instanciation

Utilise un système de gardes/motifs pour limiter le nombre d'instances générées :

$\forall x : int.P(f(g(x)))$

- garde possible : $f(g(x))$

Contexte des théories :

- $f(1)$
- $g(2)$
- $\boldsymbol{f(g(a))}$

La taille du modèle envoyé aux théories influence directement le nombre d'instances générées

Algorithme CDCL(T) + Quantificateur

while *true* **do**
- $(\Gamma, \Delta, T, Conflict) \leftarrow BCP(\Gamma, \Delta, T)$
- ...
- **else if** *all variables are assigned in* Δ **then**
  - $Instances \leftarrow instanciate(\Delta, T)$
  - **if** $Instances \neq \emptyset$ **then**
    - $\Gamma \leftarrow \Gamma \cup to\_cnf(Instances)$
  - **else**
    - **return** Unknown
- **else**
  - ...

Gestion du modèle

- Mise en CNF
- Solveur SAT par méthode des tableaux

Forme Normale Conjonctive

$$(X_1 \wedge Y_1) \vee (X_2 \wedge Y_2) \vee ... \vee (X_{n-1} \wedge Y_{n-1}) \vee (X_n \wedge Y_n)$$

Besoin de travailler sur une CNF pour un BCP performant

Forme Normale Conjonctive

$$(X_1 \land Y_1) \lor (X_2 \land Y_2) \lor ... \lor (X_{n-1} \land Y_{n-1}) \lor (X_n \land Y_n)$$

Besoin de travailler sur une CNF pour un BCP performant

$$\begin{aligned} & (X_1 \lor X_2 \lor ... \lor X_{n-1} \lor X_n) \\ \land & (X_1 \lor X_2 \lor ... \lor X_{n-1} \lor Y_n) \\ \land & ... \\ \land & (Y_1 \lor Y_2 \lor ... \lor Y_{n-1} \lor Y_n) \end{aligned}$$

- Solution exponentielle : 2^n clauses

Mise en Forme Normale Conjonctive

$$(X_1 \wedge Y_1) \vee (X_2 \wedge Y_2) \vee \ldots \vee (X_{n-1} \wedge Y_{n-1}) \vee (X_n \wedge Y_n)$$

Mise en Forme Normale Conjonctive

$$(X_1 \wedge Y_1) \vee (X_2 \wedge Y_2) \vee \dots \vee (X_{n-1} \wedge Y_{n-1}) \vee (X_n \wedge Y_n)$$

Algorithme de Tseitin (1970) :

$$(Z_1 \vee Z_2 \vee \dots \vee Z_{n-1} \vee Z_n)$$
$$\forall i.(Z_i \iff X_i \wedge Y_i)$$

Mise en Forme Normale Conjonctive

$$(X_1 \wedge Y_1) \vee (X_2 \wedge Y_2) \vee \dots \vee (X_{n-1} \wedge Y_{n-1}) \vee (X_n \wedge Y_n)$$

Algorithme de Tseitin (1970) :

$$(Z_1 \vee Z_2 \vee \dots \vee Z_{n-1} \vee Z_n)$$
$$\forall i.(Z_i \iff X_i \wedge Y_i)$$

$$(\neg Z_i \vee X_i)$$
$$(\neg Z_i \vee Y_i)$$
$$(Z_i \vee \neg X_i \vee \neg Y_i)$$

- Solution linéaire

Mise en Forme Normale Conjonctive

$$(X_1 \wedge Y_1) \vee (X_2 \wedge Y_2) \vee ... \vee (X_{n-1} \wedge Y_{n-1}) \vee (X_n \wedge Y_n)$$

Modèle booléen possible : $\{X_1; Y_1\}$

Après mise en CNF toutes les variables doivent être assignées

- Impacte le nombre de termes envoyés aux théories
- Augmente le nombre de formules quantifiées et le contexte des théories envoyé au moteur d'instanciation

Solveur SAT par méthode des tableaux d'Alt-Ergo

Solveur SAT par méthode des tableaux d'Alt-Ergo

On travaille ici directement sur la formule sans mise en CNF :

$$\overbrace{(A \vee (B \wedge C)) \wedge (D \vee ((E \vee F) \wedge G))}^{\text{x}}$$

Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \vee (B \wedge C))}_{X_1} \wedge \underbrace{(D \vee ((E \vee F) \wedge G))}_{X_2}}^{X}$$

Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{\mathsf{X}}$$

Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee (\overbrace{\underbrace{(E \vee F)}_{X_5} \wedge G}^{X_4}))}_{X_2}}^{X}$$

$[X$

Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X_4})}_{X_2}}^{X}$$

$[X; X_1; X_2]$

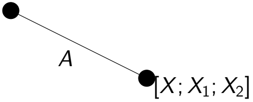
Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee (\overbrace{\underbrace{(E \vee F)}_{X_5} \wedge G}^{X_4}))}_{X_2}}^{X}$$

$[X; X_1; X_2]$

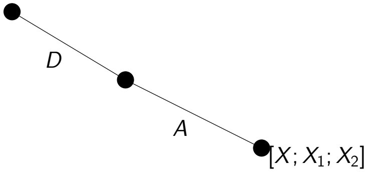
Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X_4})}_{X_2}}^{X}$$



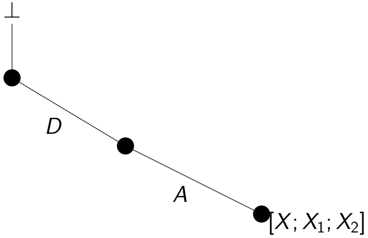
Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \lor \overbrace{(B \land C)}^{x_3})}_{x_1} \land \underbrace{(D \lor \overbrace{(\underbrace{(E \lor F)}_{x_5} \land G)}^{x_4})}_{x_2}}^{x}$$



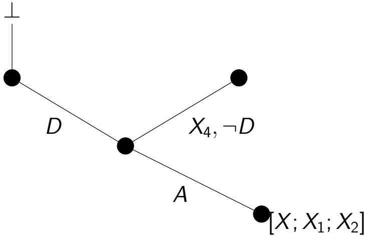
Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X_4})}_{X_2}}^{X}$$



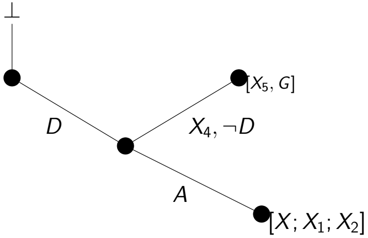
Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X_4})}_{X_2}}^{X}$$



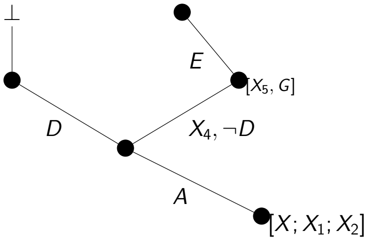
Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee (\overbrace{\underbrace{(E \vee F)}_{\color{blue}{X_5}} \wedge \color{blue}{G}}^{X_4}))}_{X_2}}^{X}$$



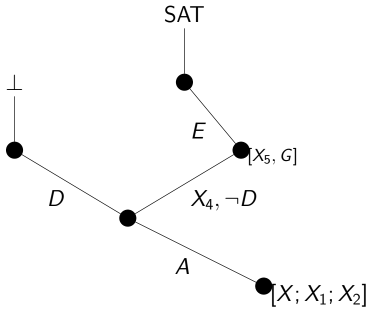
Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{\color{blue}{X_5}} \wedge G)}^{X_4})}_{X_2}}^{X}$$



Solveur SAT par méthode des tableaux d'Alt-Ergo

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{x_3})}_{x_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{x_5} \wedge G)}^{x_4})}_{x_2}}^{x}$$



Modèle booléen renvoyé par le solveur SAT : $\{A; \neg D; E; G\}$

X_i n'ont de l'influence que sur la partie SAT

Comment obtenir le meilleur d'un CDCL et de la méthode des tableaux ?

CDCL moderne :

- Efficace sur le raisonnement booléen (BCP, backjumping, apprentissage)
- Modèle total
  - Modèle booléen du solveur SAT : $\{A; B; \neg C; \neg D; \neg E; F; G\}$

Comment obtenir le meilleur d'un CDCL et de la méthode des tableaux ?

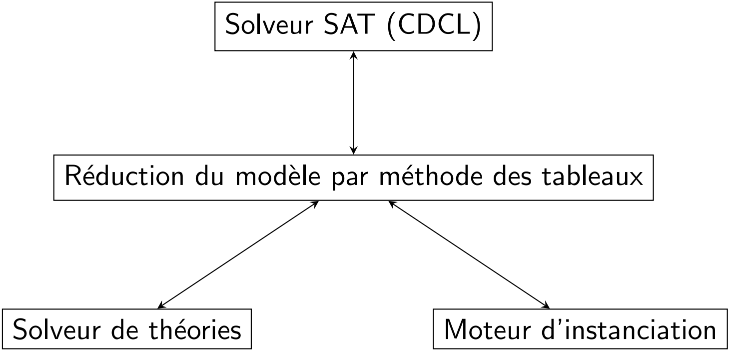
CDCL moderne :

- Efficace sur le raisonnement booléen (BCP, backjumping, apprentissage)
- Modèle total
  - Modèle booléen du solveur SAT : $\{A; B; \neg C; \neg D; \neg E; F; G\}$

Méthode des Tableaux :

- Pas efficace sur le raisonnement booléen (pas d'apprentissage)
- Modèle partiel
  - Modèle booléen du solveur SAT : $\{A; \neg D; E; G\}$

Nouvelle architecture d'Alt-Ergo



- Parcourir la Formule de base
- Utilisant les décisions et propagations du CDCL

CDCL(*Tableaux*(*T*))

while *true* **do**
  …
  $(T, Conflict) \leftarrow theory\_assume(tableaux\_reduce(\Delta), T)$
  …
  else if *all variables are assigned in* Δ **then**
    $Instances \leftarrow instanciate(tableaux\_reduce(\Delta), T)$
    if $Instances \neq \emptyset$ **then**
      $\Gamma \leftarrow \Gamma \cup to\_cnf(Instances)$
    else
      return Unknown
  else
    …

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{x_3})}_{x_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{x_5} \wedge G)}^{x4})}_{x_2}}^{x}$$

Modèle booléen partiel du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{X}$$

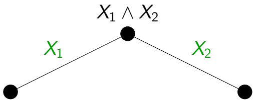
$X_1 \wedge X_2$

Modèle booléen partiel du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{X}$$

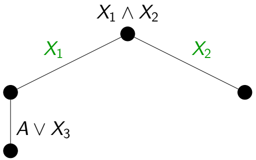


Modèle booléen partiel du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee (\overbrace{\underbrace{(E \vee F)}_{X_5} \wedge G}^{X4}))}_{X_2}}^{X}$$

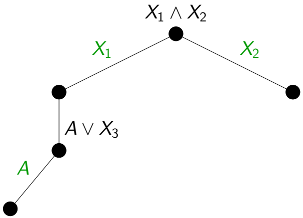


Modèle booléen partiel du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{\mathrm{X}}$$

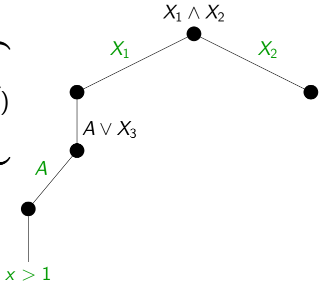


Modèle booléen partiel du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{x4})}_{X_2}}^{\mathrm{x}}$$

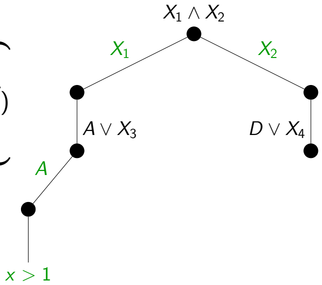


Modèle booléen partiel du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{\mathrm{X}}$$

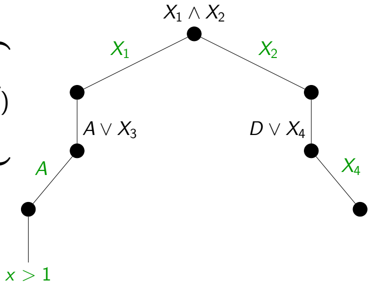


Modèle booléen partiel du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{\mathrm{x}}$$

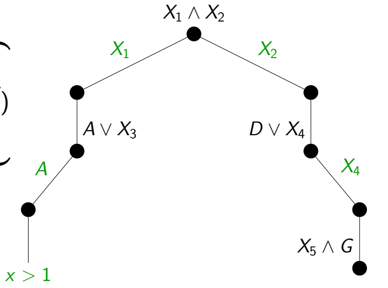


Modèle booléen partiel du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{\mathrm{X}}$$

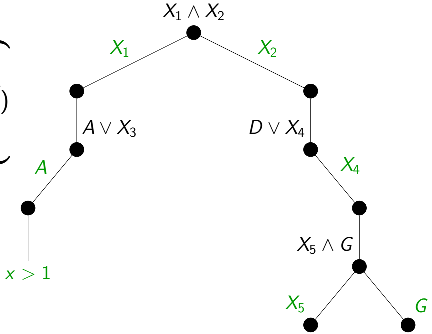


Modèle booléen partiel du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee (\overbrace{\underbrace{(E \vee F)}_{X_5} \wedge G}^{X4}))}_{X_2}}^{\mathrm{x}}$$



Modèle booléen partiel du solveur SAT :

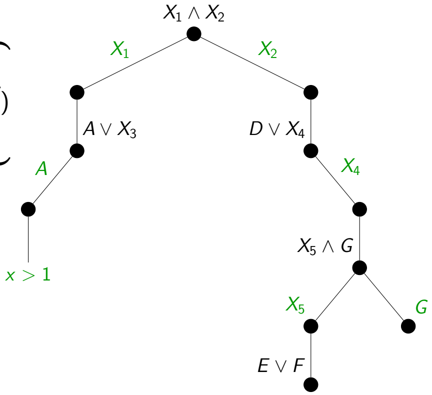
$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \lor \overbrace{(B \land C)}^{X_3})}_{X_1} \land \underbrace{(D \lor \overbrace{(\underbrace{(E \lor F)}_{X_5} \land G)}^{X4})}_{X_2}}^{\mathrm{X}}$$

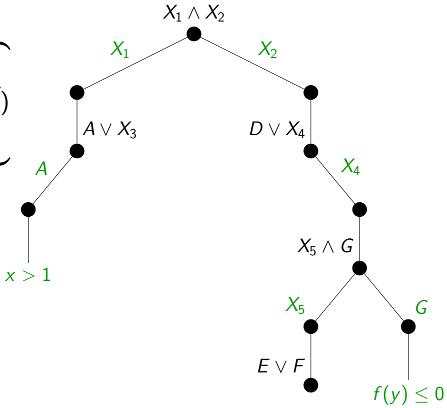
Modèle booléen partiel du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$



Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{X}$$



Modèle booléen partiel du solveur SAT :

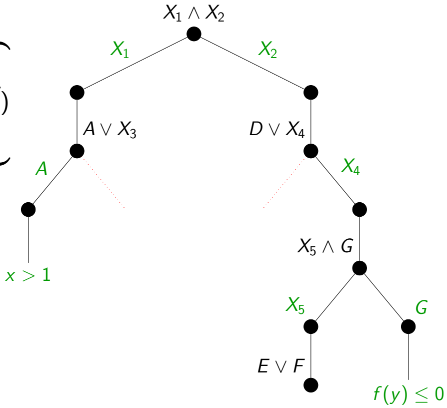
$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{\mathrm{X}}$$

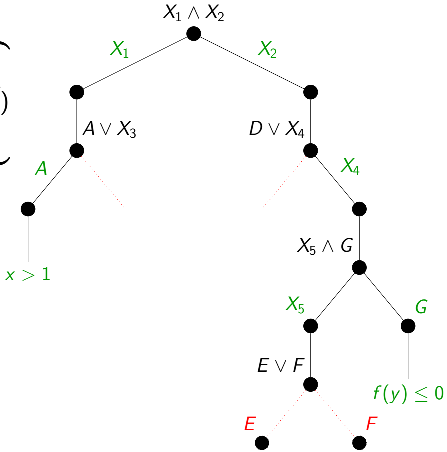
Modèle booléen partiel du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$E, F, C = -$$



Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{\mathrm{X}}$$



Modèle booléen partiel du solveur SAT :

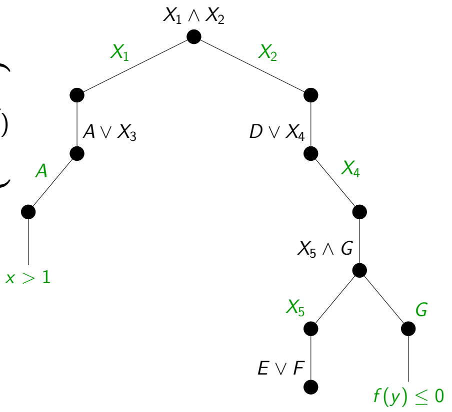
$$A, G = \top$$

$$B, D = \bot$$

$$E, F, C = -$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{X}$$



Modèle booléen complet du solveur SAT :

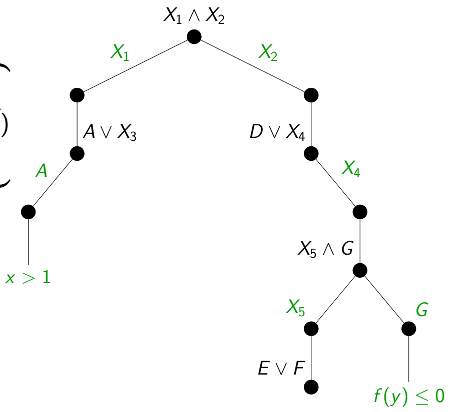
$$A, G = \top$$
$$B, D = \bot$$
$$C = \bot$$
$$E, F = \top$$

Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{X}$$

Modèle booléen complet du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$C = \bot$$
$$E, F = \top$$

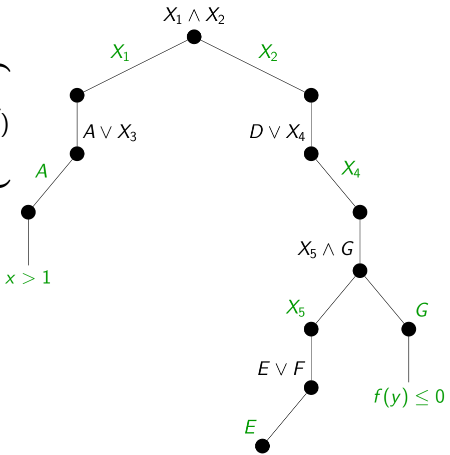


Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \vee \overbrace{(B \wedge C)}^{X_3})}_{X_1} \wedge \underbrace{(D \vee \overbrace{(\underbrace{(E \vee F)}_{X_5} \wedge G)}^{X4})}_{X_2}}^{x}$$

Modèle booléen complet du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$C = \bot$$
$$E, F = \top$$

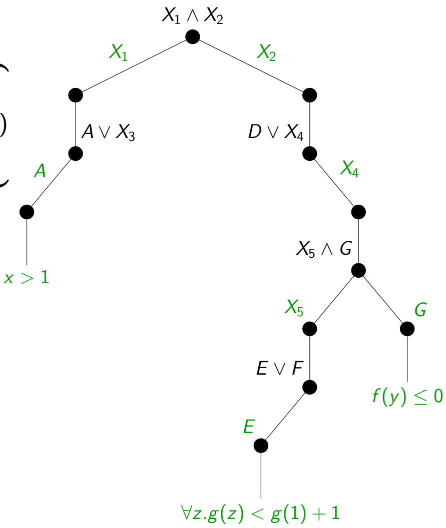


Réduction du modèle par méthode des tableaux

$$\overbrace{\underbrace{(A \lor \overbrace{(B \land C)}^{X_3})}_{X_1} \land \underbrace{(D \lor \overbrace{(\underbrace{(E \lor F)}_{X_5} \land G)}^{X4})}_{X_2}}^{X}$$

Modèle booléen complet du solveur SAT :

$$A, G = \top$$
$$B, D = \bot$$
$$C = \bot$$
$$E, F = \top$$



Réduction du modèle par méthode des tableaux

$$(A \vee (B \wedge C)) \wedge (D \vee ((E \vee F) \wedge G))$$

Modèle booléen complet : $\{A; \neg B; \neg C; \neg D; E; F; G\}$

Modèle réduit : $\{A; E; G\}$

CDCL(*Tableaux*(*T*))

Résultats

CDCL(*Tableaux*(*T*))

Résultats

	# buts	cdcl	tableaux	cdcl + tableaux
BWARE–DAB	860	98.7% (258s)	100% (417s)	**100%(47s)**
BWARE–RCS3	2256	98.7% (742s)	98.9% (685s)	**99.0%(725s)**
BWARE–p4	9341	98.4% (2097s)	99.3% (2279s)	**99.4(790s)**
BWARE–p9	371	64.7% (1104s)	67.9% (342s)	**72.2%(492s)**
EACSL	959	75.6% (64s)	**93.3% (258s)**	92.3%(293s)
SPARK	16773	80.5% (1769s)	83.6% (2757s)	**84.0%(2298s)**
WHY3	2003	38.9% (616s)	**72.0% (1876s)**	69.8% (1471s)
Total	32563	84.5% (6652s)	89.0% (8617s)	**89.1%(6119s)**

CDCL(*Tableaux*(*T*))

Résultats

	# buts	`cdcl`	`tableaux`	`cdcl + tableaux`
BWARE–DAB	860	98.7% (258s)	100% (417s)	**100%(47s)**
BWARE–RCS3	2256	98.7% (742s)	98.9% (685s)	**99.0%(725s)**
BWARE–p4	9341	98.4% (2097s)	99.3% (2279s)	**99.4(790s)**
BWARE–p9	371	64.7% (1104s)	67.9% (342s)	**72.2%(492s)**
EACSL	959	75.6% (64s)	**93.3% (258s)**	92.3%(293s)
SPARK	16773	80.5% (1769s)	83.6% (2757s)	**84.0%(2298s)**
WHY3	2003	38.9% (616s)	**72.0% (1876s)**	69.8% (1471s)
Total	32563	84.5% (6652s)	89.0% (8617s)	**89.1%(6119s)**

- Solution efficace (-29% en temps)
- Permet à Alt-Ergo d'être performant sur des problèmes fortement booléens
- Tout en restant performant sur des problèmes quantifiés

Extension du standard SMT-LIB2 avec du polymorphisme

Alt-Ergo et le standard SMT-LIB2

Alt-Ergo :

- Ne supporte pas le standard SMT-LIB2
- Impossibilité d'accéder à la communauté et à ses bancs de tests

SMT-LIB2

- Ne supporte pas le polymorphisme
- Besoin de monomorphiser (indécidable)
- Dégrade les gardes des formules quantifiées

Le langage natif d'Alt-Ergo

- Programme généré par l'outil Why3

```
type 'a option

logic None : 'a option
logic Some : 'a -> 'a option
logic match_option : 'a option, 'b, 'b -> 'b

axiom match_option_None :
(forall z:'a. forall z1:'a. (match_option((None : 'b option), z, z1) = z))

axiom match_option_Some :
(forall z:'a. forall z1:'a. forall u:'b. (match_option(Some(u), z,z1) = z1))
```

Le standard SMT-LIB2

- Programme monomorphisé

```
(declare−sort uni 0)
(declare−sort ty 0)
(declare−fun sort (ty uni) Bool)
(declare−fun option (ty) ty)

(declare−fun None (ty) uni)
(assert (forall ((a ty)) (sort (option a) (None a))))
(declare−fun Some (ty uni) uni)
(assert (forall ((a ty)) (forall ((x uni)) (sort (option a) (Some a
     x)))))

(declare−fun match_option (ty ty uni uni uni) uni)
(assert (forall ((a ty) (a1 ty)) (forall ((x uni) (x1 uni) (x2 uni))
          (sort a1 (match_option a1 a x x1 x2)))))
...
```

Extension de la syntaxe du standard SMT-LIB2

- Rafraîchissement des travaux de la communauté
- Notation (*par* (*A*) ..) dont l'usage était restreint

Extension de la syntaxe du standard SMT-LIB2

- Rafraîchissement des travaux de la communauté
- Notation (*par* (*A*) ..) dont l'usage était restreint

```
(declare-sort option 1)

(declare-const None (par (A) (option A)))
(declare-fun Some (par (A) (A) (option A)))
(declare-fun match_option (par (B A) ((option A) B B)
    B))

;; match_option_None
(assert (par (B A) (forall ((z B) (z1 B))
          (= (match_option (as None (option A)) z z1)
    z))))

;; match_option_Some
(assert (par (B A) (forall ((z B) (z1 B) (u A))
          (= (match_option (Some u) z z1) z1))))
```

Extension polymorphe conservative du standard SMT-LIB2

- Bibliothèque standalone en OCaml
  - Parsing et typage de la SMT-LIB2
  - Et de son extension polymorphe
- Intégration de la bibliothèque à Alt-Ergo, lui permettant de supporter des fichiers au format du standard.
  - remplacer le langage natif par le standard SMT-LIB2 étendu

Impact du polymorphisme

- Banc de tests issus de la preuve de programme aux formats SMT-LIB2, SMT-LIB2 polymorphe et natif d'Alt-Ergo.
  - smt2 : 86.8% (15173s)
  - smt2 polymorphe: 89.1% (6656s)
  - natif : **89.1% (6119s)**

Alt-Ergo et les solveurs de l'état de l'art

Fichiers identiques au format SMT-LIB2

- Transformations : monomorphisation, `let-in`, `ite`, bitvector

Smt2 commun	#buts	`Alt-ergo`	`CVC4`	`vampire`	`Z3`
BWARE–DAB	860	**88.9% (712s)**	59.5% (499s)	40.6%(3638s)	84.0%(110s)
BWARE–RCS3	2256	**98.4% (1746s)**	90.4% (2056s)	22.7% (7304s)	96.1%(363s)
BWARE–p4	9341	94.1% (8186s)	**96.8% (7723s)**	20.4%(59618s)	84.3% (3994s)
BWARE–p9	371	60.1% (826s)	**70.1%(1560s)**	4.8% (305s)	64.2% (505s)
EACSL	959	**90.7% (476s)**	71.2% (50s)	45.2% (585s)	70.8% (202s)
SPARK	16773	**83.9%(1881s)**	83.5% (612s)	77.6% (3351s)	82.4% (324s)
WHY3	2003	**65.9% (1343s)**	60.2% (650s)	40.7% (1453s)	58.9%(2331s)
Total	32563	**86.8% (15173s)**	85.2% (13154s)	51.8% (76257s)	81.9% (7832s)

Compétition SMT (2018)

- Choix des catégories proches de la preuve de programme
  - plusieurs théories
  - avec quantificateurs universels

Compétition SMT (2018)

- Choix des catégories proches de la preuve de programme
  - plusieurs théories
  - avec quantificateurs universels

2018	AUFNIRA	AUFLIRA	ALIA
Alt-Ergo	4 (69.2%)	3 (98.2%)	3 (92.8%)
CVC4	**1 (72.4%)**	2 (98.8%)	2 (95.2%)
Vampire	2 (68.9%)	4 (97.6%)	4 (64.3%)
veriT	-	5 (96.5%)	4 (64.3%)
z3	3 (69.7%)	**1 (99.2%)**	**1 (100%)**
Total	4 (1480)	5 (20011)	5 (42)

Compétition SMT (2018)

- Choix des catégories proches de la preuve de programme
  - plusieurs théories
  - avec quantificateurs universels

2018	AUFNIRA	AUFLIRA	ALIA
Alt-Ergo	4 (69.2%)	3 (98.2%)	3 (92.8%)
CVC4	**1 (72.4%)**	2 (98.8%)	2 (95.2%)
Vampire	2 (68.9%)	4 (97.6%)	4 (64.3%)
veriT	-	5 (96.5%)	4 (64.3%)
z3	3 (69.7%)	**1 (99.2%)**	**1 (100%)**
Total	4 (1480)	5 (20011)	5 (42)

- SMTCOMP 2019
  - Plus de catégories
  - Catégories sans quantificateurs

Conclusions

Conclusions

- Intégration d'un solveur SAT efficace au solveur SMT Alt-Ergo
  - Gain de 29% du temps d'exécution par rapport au solveur historique

Conclusions

- Intégration d'un solveur SAT efficace au solveur SMT Alt-Ergo
  - Gain de 29% du temps d'exécution par rapport au solveur historique
- Extension du standard SMT-LIB 2 avec du polymorphisme
  - Création d'une bibliothèque supportant le standard SMT-LIB2 étendu

Conclusions

- Intégration d'un solveur SAT efficace au solveur SMT Alt-Ergo
  - Gain de 29% du temps d'exécution par rapport au solveur historique
- Extension du standard SMT-LIB 2 avec du polymorphisme
  - Création d'une bibliothèque supportant le standard SMT-LIB2 étendu
- Participation à la compétition SMT

Conclusions

- Intégration d'un solveur SAT efficace au solveur SMT Alt-Ergo
  - Gain de 29% du temps d'exécution par rapport au solveur historique
- Extension du standard SMT-LIB 2 avec du polymorphisme
  - Création d'une bibliothèque supportant le standard SMT-LIB2 étendu
- Participation à la compétition SMT

Perspectives :

- Utiliser un solveur SAT de l'état de l'art
- Supporter les vecteurs de bits
- Effectuer du pré-traitement

Conclusions

- Intégration d'un solveur SAT efficace au solveur SMT Alt-Ergo
  - Gain de 29% du temps d'exécution par rapport au solveur historique
- Extension du standard SMT-LIB 2 avec du polymorphisme
  - Création d'une bibliothèque supportant le standard SMT-LIB2 étendu
- Participation à la compétition SMT

Perspectives :

- Utiliser un solveur SAT de l'état de l'art
- Supporter les vecteurs de bits
- Effectuer du pré-traitement

Merci pour votre attention